CHAIR OF DECENTRALIZED INFORMATION SYSTEMS & DATA MANAGEMENT

TECHNISCHE UNIVERSITY MUNICH

Thesis proposal

Scalabe range lock

Author: **Thua Duc Nguyen**

Supervisor: **Prof. Dr. Viktor Leis**

Advisor: **Lam Duy Nguyen**

1 Introduction

A scalable range lock is a synchronization construct designed to address the problem of multiple writers attempting to modify different parts of the same file. Unlike conventional methods that rely on a singular lock for the entire file, range locks enables concurrent access to disjoint parts of a shared resource, thus allow parallelism for writers working on different parts of the same file and solve the bottleneck of using singular lock.

2 Motivation

Recently, there has been an increase in interest in range locking techniques [1, 2, 3], with a notable example being the Linux kernel community's consideration of using range locking techniques to replace the bottleneck of the `mmap_lock`. The `mmap_lock` is a reader-writer semaphore that controls access to the whole `mm_struct`[4] and protect VMAs (Virtual memory area). Because this lock cover the whole `mm_struct`, it does not allow to have paralell update and page fault, event when it happend in different VMAs. It is therefore a coarse-grained lock that creates contention in the memory management subsystem. There have been several attempts to overcome the scalability issues of `mmap_lock`, but the problem is far from solved[3].

In the context of database management systems, range locks could be beneficial in scenarios involving large databases and indexes. For instance, the standard hierarchical locking approach locks the entire index or an index partition before locking individual pages or keys. However, with the exponential growth in data sizes, locking at the index level may become too coarse-grained for concurrent transactions[5]. This can lead to significant memory overhead and performance bottlenecks associated with acquiring and releasing locks. Range locks offer a solution to this problem by focusing on key ranges rather than entire indexes. This reduces contention, enhances concurrency and allows multiple transactions to operate concurrently on disjoint key ranges.

3 Related Work

Previous research has explored various approaches to a scalable range lock. The current implementation of area locking in the kernel uses an interval tree, which keeps track of the areas that have been acquired and requested, and an internal spin lock to protect it. This spin lock becomes a point of contention on its own when the range lock is frequently acquired[6].

A different approach to range locking is to build it based on a liked list instead of a range tree, where each node represents an acquired range[6]. Although using a list makes it archivable to maintain lock-less fashion, inserting and searching the list might be less efficient.

4 Approach

The proposed approach involves designing and implementing ab efficient scalable range locking mechanism. The key components of the approach include:

- Designing an algorithm that safely and efficiently locks arbitrary ranges of a shared resources.
- Approaching the problem with both interval tree and linked-list approaches to find the optimize solution.
- Developing a flexible and easy-to-use API for integrating the range lock into existing distributed systems seamlessly.

5 Evaluation

The proposed approach will be evaluated through extensive experimentation and benchmark under various workload scenarios. The evaluation criteria include:

- Performance: It is necessary to measure the performance and throughput of the range locking mechanism under increasing load and concurrent accesses.
- Correctness: It is also important to validate the consistency and correctness of data accesses, especially in scenarios involving overlapping data ranges and concurrent operations.
- Scalability: The scalability limits of the range locking mechanism must be assessed in terms of the number of readers and writers, data range sizes, and system resources.
- Comparison: A comparison must be made between the performance and scalability of the proposed mechanism and those of existing locking schemes and state-of-the-art approaches.

6 Expected Outcome

The expected outcome of this research is a scalable range locking mechanism that significantly improves the efficiency and scalability of the existed range locks. The findings from the evaluation will provide insights into the performance characteristics, scalability limits, and potential trade-offs of the proposed mechanism, contributing to the advancement of distributed computing research and practice.

7 Resources

The following resources are required to evaluate the proposed mechanism:

- Budget of 32 cores and 32GB of RAM for one month for experimentation and benchmarking.

Resources are critical when evaluating performance under multi-threaded conditions. With such resources, thorough evaluations can be performed to understand how the region locking mechanism performs under heavy contention. Utilizing all available cores and memory allows the implementation to be pushed to its limits and gain insight into its behavior in complex multi-threaded environments.

Bibliography

[1] J. Corbet. "Range reader/writer locks for the kernel". In: *LWN.net* (2022). Accessed: 2024-04-21. URL: https://lwn.net/Articles/724502/.

[2] M. Rybczynska. "Introducing maple trees". In: *LWN.net* (2022). Accessed: 2024-04-21. URL: https://lwn.net/Articles/845507/.

[3] J. Corbet. "The ongoing search for mmap_lock scalability". In: *LWN.net* (2022). Accessed: 2024-04-21. URL: https://lwn.net/Articles/893906/.

[4] S. Boutnaru. "Introducing maple trees". In: *medium.com* (2023). Accessed: 2024-04-21. URL: Linux%20Kernel%20%E2%80%94%20mm%5C_struct.

[5] G. Graefe. "Hierarchical Locking in B-Tree Indexes". In: 2020. URL: https://dl.gi.de/server/api/core/bitstreams/9f283897-1a24-4714-aa57-c773e6c2da96/content.

[6] A. K. D. Dice and S. Issa. *Scalable range locks for scalable address spaces and beyond.* Accessed: 2024-04-21. 2020. URL: https://arxiv.org/pdf/2006.12144.pdf.